Amendments to the Claims

Please cancel claim 5 without prejudice.

1. (currently amended) A computer-implemented method for processing numerical values in a computer program executable on a computer system, comprising:

encapsulating in a large-integer datatype, large-integer data and associated operators, wherein the large-integer data has runtime expandable precision and maximum precision is limited only by system memory availability; ~~and~~

overloading language-provided arithmetic, logical, and type conversion operators with the large-integer operators that operate on large-integer variables in combination with other datatypes, and programmed usage of a variable of the large-integer datatype is equivalent to and interoperable with a variable of a system-defined integral datatype;

establishing a plurality of available storage nodes available for allocation to large-integer data; and

allocating a subset of the plurality of available storage nodes for a large-integer variable, the subset being an allocated plurality of storage nodes, and storing a numerical value in the allocated plurality of storage nodes and forming a linked list of the allocated plurality of storage nodes.

2. (original) The method of claim 1, further comprising converting a character string into large-integer data in response to a constant definition statement.

3. (original) The method of claim 2, further comprising converting large-integer data to and from a character string for input, output, and serialization.

4. (original) The method of claim 1, further comprising:

converting input data from language-provided input functions to large-integer data; and

converting large-integer data to a format compatible with language-provided output functions.

5. (canceled)

6. (currently amended) The method of claim 1 ~~5~~, further comprising allocating a selected number of bits for each storage node in response to a program-specified parameter.

7. (currently amended) The method of claim 1 ~~5~~, further comprising dynamically allocating a number of storage nodes for storage of the numerical value as a function of a size of the numerical value.

8. (currently amended) The method of claim 7, further comprising storing in each node that is allocated to a large-integer variable, a subset of bit values that represent a numerical value.

9. (original) The method of claim 8, further comprising:

maintaining a set of available storage nodes that are not allocated to any large-integer variable;

allocating a storage node from the set of available storage nodes to a large-integer variable while performing a large-integer operation that generates a numerical value and stores the numerical value in the variable, if a number of bit values required to represent the numerical value exceeds storage available in storage nodes allocated to the large-integer variable; and

returning to the set of available storage nodes a storage node allocated to a large-integer variable while performing a large-integer operation that generates a numerical value for storage in the variable, if a number of bit values required to represent the numerical value is less than storage available in storage nodes allocated to the variable.

10. (original) The method of claim 9, further comprising overloading language-provided memory allocation and deallocation operators with large-integer operators that allocate and deallocate storage nodes.

11. (original) The method of claim 1, further comprising, responsive to a large-integer divide operation specifying an input dividend and divisor:

identifying a set of most-significant bits of the dividend and a set of least-significant bits of the dividend;

recursively performing a large-integer divide operation using the set of most-significant bits as the input dividend, and returning a quotient and a remainder;

finding a lower-part dividend as a function of the remainder and the set of least-significant bits;

recursively performing a large-integer divide operation using the lower-part dividend; and

concurrently solving for the quotient and the remainder.

12. (original) The method of claim 11, further comprising identifying an optimal set of most-significant bits of the dividend and a set of least-significant bits of the dividend as a function of a number of bits that represent the dividend and a number of bits that represent the divisor.

13. (currently amended) The method of claim 12, further comprising identifying an optimal set of most-significant bits of the dividend and a set of least-significant bits of the dividend as a function <u>of</u> one-half a difference between the number of bits that represent the dividend and the number of bits that represent the divisor.

14. (original) The method of claim 1, further comprising emulating fixed-bit arithmetic on variables of the large-integer data type.

15. (original) The method of claim 1, further comprising transferring data associated with temporary variables of the large-integer datatype by moving pointers to the data.

16. (original) The method of claim 1, further comprising

encapsulating in a large-floating-point datatype, large-floating-point data and associated operators, wherein the large-floating-point data has runtime expandable precision and maximum precision is limited only by system memory availability; and

overloading language-provided arithmetic, logical, and type conversion operators for floating-point data with the large-floating-point datatype operators that operate on large-floating-point variables in combination with other datatypes, and programmed usage of a variable of the large-floating-point datatype is equivalent to and interoperable with a variable of a system-defined floating-point datatype.

17. (original) The method of claim 1, further comprising

encapsulating in a large-rational datatype, large-rational data and associated operators, wherein the large-rational data has runtime expandable precision and maximum precision is limited only by system memory availability; and

overloading language-provided arithmetic, logical, and type conversion operators for rational data with the large-rational datatype operators that operate on large-rational variables in combination with other datatypes, and programmed usage of a variable of the large-rational datatype is equivalent to and interoperable with a variable of a system-defined rational datatype.

18. (currently amended) An apparatus for processing numerical values in a computer program executable on a computer system, comprising:

means for encapsulating in a large-integer datatype, large-integer data and associated operators, wherein the large-integer data has runtime expandable precision and maximum precision is limited only by system memory availability; ~~and~~

means for overloading language-provided arithmetic, logical, and type conversion operators for integers with the large-integer datatype operators that operate on large-integer variables in combination with other datatypes, and programmed usage of a variable of the large-integer datatype is equivalent to and interoperable with a variable of a system-defined integral datatype;

means for establishing a plurality of allocable storage nodes available for allocation to large-integer data;

means for allocating, for a large-integer variable, a subset of the plurality of allocable storage nodes, the subset becoming an allocated plurality of storage nodes for the large-integer variable; and

means for storing a numerical value in the allocated plurality of storage nodes and forming a linked list of the allocated plurality of storage nodes.

19. (original) The apparatus of claim 18, further comprising

means for encapsulating in a large-floating-point datatype, large-floating-point data and associated operators, wherein the large-floating-point data has runtime expandable precision and maximum precision is limited only by system memory availability; and

means for overloading language-provided arithmetic, logical, and type conversion operators for floating-point data with the large-floating-point datatype operators that operate on large-floating-point variables in combination with other datatypes, and programmed usage of a variable of the large-floating-point datatype is equivalent to and interoperable with a variable of a system-defined floating-point datatype.

20. (original) The apparatus of claim 18, further comprising

means for encapsulating in a large-rational datatype, large-rational data and associated operators, wherein the large-rational data has runtime expandable precision and maximum precision is limited only by system memory availability; and

means for overloading language-provided arithmetic, logical, and type conversion operators for rational data with the large-rational datatype operators that operate on large-rational variables in combination with other datatypes, and programmed usage of a variable of the large-rational datatype is equivalent to and interoperable with a variable of a system-defined rational datatype.

21. (new) The method of claim 1, further comprising:

determining a total number of available storage nodes available for allocation to large-integer data;

allocating memory for a first number of available storage nodes, responsive to the total number being less than first threshold value, and establishing the first number of available storage nodes; and

removing from the plurality of available storage nodes, responsive to the total number being greater than a second threshold value, a second number of storage nodes and deallocating memory for the second number of storage nodes.